Assignment Tasks | Batch-7

Task 1: Set Up SonarQube Locally Using Linux Package and PostgreSQL

1. Install SonarQube on your local machine using the official Linux package.
2. Install and configure PostgreSQL as the backend database for SonarQube.
    - Create a database named sonarqube.
    - Create a database user with appropriate permissions.
3. Configure SonarQube to connect to the PostgreSQL database.
4. Start the SonarQube service and access it on http://localhost:9000.
5. Verify that the SonarQube server is running by logging in as the default administrator (admin/admin).

Task 2: Integrate Jenkins with SonarQube

1. Install the **SonarQube Scanner** plugin in Jenkins.
2. Configure Jenkins to connect to your SonarQube server.
    - Add SonarQube credentials in Jenkins.
    - Configure Sonarqube server in System
    - Set up the SonarQube scanner in the Jenkins global tool configuration.
3. Verify the integration by running a basic SonarQube scan from a Jenkins pipeline.

Task 3: Create 3 Pipelines to Showcase SonarQube Reports

1. Create **three Jenkins pipelines** for the following types of projects:
    - **Java Project**: Use a Maven-based Java project.
    - **Node.js Project**: Use an npm-based Node.js project.
    - **Python Project**: Use a pip-based Python project.
2. For each pipeline:
    - Include stages to build the project and scan it with SonarQube & perform Quality Gate check.
    - Ensure the SonarQube analysis is successfully uploaded and visible in the SonarQube dashboard.
3. Showcase:
    - **Code quality metrics**.

- **Code smells**, **vulnerabilities**, and **bugs**.
- **Code Coverage**

Task 4: Ensure Code Coverage Visibility

1. Integrate a testing framework with each project to measure code coverage:
   - **Java**: Use JaCoCo for code coverage.
   - **Node.js**: Use nyc or jest for code coverage.
   - **Python**: Use coverage.py or pytest-cov for code coverage.
2. Ensure the **code coverage percentage** is visible in the SonarQube dashboard for each project.

Task 5: Perform Analysis on Different Branches

1. Setup SonarQube with **Community Branch Plugin** to enable branch analysis.
2. Create at least two branches for each project (e.g., main and feature/new-feature).
3. Perform a SonarQube analysis on both branches

Task 6: Set Up SonarQube Using Docker Container

1. Install Docker and Docker Compose on your local machine.
2. Set up SonarQube with Docker using the official sonarqube image.
3. Access SonarQube on http://localhost:9000 and verify the setup.

Task 7: Set Up SonarQube Locally with SSL Configuration

1. Generate SSL certificates (self-signed or from a certificate authority).
2. Configure SonarQube to use SSL certificates.
   - Update SonarQube's sonar.properties file to enable HTTPS.
3. Access SonarQube on an HTTPS URL (https://localhost:9000) and verify the secure connection.
4. Document the process, including steps to generate and configure the certificates.

Submission Requirements

1. **Documentation**:
    - Detailed steps for each task.
    - Screenshots showing successful execution and results (e.g., Jenkins pipelines, SonarQube reports).

Evaluation Criteria

Task	**Criteria**
Task 1: SonarQube Setup	Successful setup with PostgreSQL backend.
Task 2: Jenkins Integration	Correct integration and pipeline execution.
Task 3: Project Pipelines	Pipelines for all 3 projects, successful analysis, and visible reports.
Task 4: Code Coverage	Accurate code coverage visible for all projects.
Task 5: Branch Analysis	Multiple branches analyzed and differences showcased in the SonarQube UI.
Task 6: Docker Setup	Properly configured and functional Docker-based SonarQube setup.
Task 7: SSL Configuration	SSL-enabled SonarQube accessed over HTTPS.